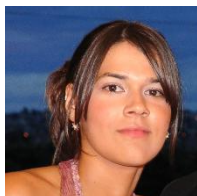


Tags [Definidas pelo Editor]

# Plugin para a contagem de pontos por caso de uso



**Andréia Alencar Carvalho da Silva**

deia.alencar@hotmail.com

Estudante de Engenharia da Computação na Universidade Tecnológica Federal do Paraná – Campus Cornélio Procopio.

<< demais autores >>

[A partir daqui inicia a parte **TEÓRICA** do artigo]

<b>De que se trata o artigo</b>
Esse artigo trata de um <i>plugin</i> desenvolvido para realizar o cálculo dos pontos por caso de uso, uma técnica de estimativa de software, em um projeto <i>astah</i> . O artigo explica o que é a técnica, como utilizá-la na teoria e como utilizar o <i>plugin</i> para realizar esse cálculo em um projeto UML desenvolvido utilizando a ferramenta <i>astah</i> .
<b>Em que situação o tema é útil</b>
Quando um usuário necessita saber qual a estimativa de esforço para o desenvolvimento de um determinado software através de seus diagramas UML, esse artigo é muito útil, pois trata de uma ferramenta para realizar os cálculos dessa estimativa, sem muito esforço do usuário.

[Plugin para a contagem de pontos por caso de uso]

Neste artigo, você poderá ter noções básicas de diagrama de caso de uso, de classe e de sequência, com ilustrações exemplificando cada um deles. Noções de métricas de software, o que são e quais os tipos existentes. Como desenvolver *plug-ins* para a ferramenta *astah*, como instalá-los e utilizá-los também, exemplificado com ilustrações. E, também, uma explicação teórica e prática do que são pontos por caso de uso, e a utilização de *plug-in* para aplicação do cálculo de pontos por caso de uso no *astah*.

Atualmente poucas empresas brasileiras produzem software com padrões de qualidade reconhecidos por algum modelo. Esta afirmação pode ser constatada ao ser analisada a quantidade de empresas certificadas no Modelo Referencial baseado em Maturidade, Capacidade e Integração (CMMI), CMMI-DEV e CMMI-SVC, o Brasil possui 221 empresas certificadas. Para efeitos comparativos, a China possui 3.316, os Estados Unidos da América 2186 e a Índia 959.

Este fato, aliado à alta carga tributária e a deficiência na formação da mão de obra na área de tecnologia da informação, mais precisamente na área de engenharia de software, caracterizam-se como fonte inibidora no processo de expansão externa do Brasil neste setor. Universidades, empresas e governo devem desenvolver mecanismos que alterem este cenário.

O Standish Group International (2013), mostra em seus estudos que, a atividade de gestão de projetos é uma importante ferramenta para as empresas produtoras de software. Ao analisar os números, vide Tabela 1, apresentados pelo Standish Group International, é possível constatar que aproximadamente dois terços dos projetos de software não obtiveram o sucesso esperado.

	2004	2006	2008	2010	2012
Sucesso	29%	35%	32%	37%	39%
Falharam	18%	19%	24%	21%	18%
Mudaram	53%	46%	44%	42%	43%

**Tabela 1: Situação de Desenvolvimento de Projeto de Software de 2004 a 2012**

Fonte: CHAOS MANIFESTO 2013, Think Big, Act Small, p.1

Dentro deste contexto é possível afirmar que a gestão de projetos de software pode ser considerado uma atividade de extrema importância e pode ajudar a garantir a excelência, qualidade e confiabilidade no desenvolvimento de projetos. Garantir essas características no desenvolvimento de um projeto requer estimativas de esforço (número de pessoas e hora de trabalho), custo e prazo.

Para estimar o esforço e o prazo associados ao desenvolvimento de projetos existem as medidas de tamanho de software, entre estas medidas é possível citar os pontos por função e os pontos por caso de uso.

Com base nos dados quantitativos apresentados, esse artigo tem como objetivo apresentar o desenvolvimento de um *plugin* para a contagem de pontos por caso de uso, que será embutido na ferramenta astah professional (veja a sessão links). Informações sobre o desenvolvimento, a instalação do *plugin* na ferramenta e a sua utilização em um exemplo, também são caracterizados neste trabalho.

## Noções básicas de diagrama de caso de uso, de classe e de sequência.

Um diagrama de caso de uso descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema. Nesse diagrama não nos aprofundamos em detalhes técnicos que dizem como o sistema faz.

Diagramas de Casos de Uso são compostos basicamente por quatro partes:

- **Cenário:** Sequência de eventos que acontecem quando um usuário interage com o sistema.
- **Ator:** Usuário do sistema, ou melhor, um tipo de usuário.
- **Caso de uso:** É uma tarefa ou uma funcionalidade realizada pelo ator (usuário).
- **Comunicação:** Tem como objetivo unir um ator com um caso de uso.

O diagrama de classes representa a estrutura do sistema, recorrendo ao conceito de classe e suas relações. O modelo de classes resulta de um processo de abstração onde são identificados os objetos relevantes do sistema em estudo. Um objeto é uma ocorrência que tem interesse para o sistema e que pretende-se descrever no seu ambiente, contendo identidade e comportamento. O comportamento de um objeto define o modo como ele age e reage a estímulos externos e a identidade de um objeto é um atributo que o distingue de todos os demais, sendo preservada quando o seu estado muda. Um objeto não é mais do que uma instância da classe. Os objetos pertencentes a este diagrama são:

- **Classe:** é a representação de um conjunto de objetos que partilham os mesmos atributos e comportamentos.
- **Relação:** representa a ligação entre classes.

Diagrama de sequência consiste em um diagrama que tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do tempo para a realização de uma operação. Os conceitos existentes nos diagramas de sequência são:

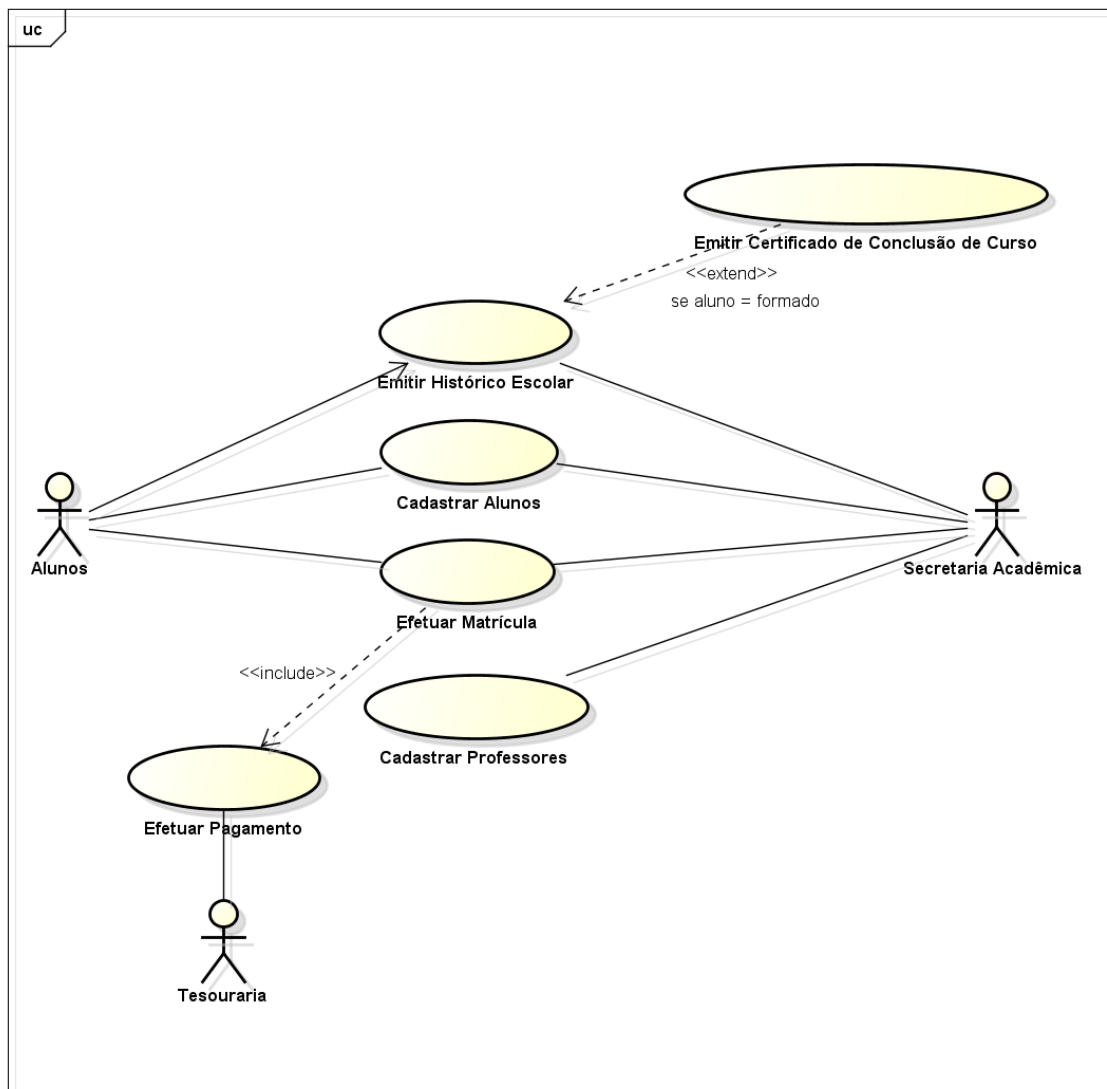
- **Atores:** São entidades externas que interagem com o sistema e que solicitam serviços. Normalmente, o ator é o responsável por enviar a mensagem inicial que inicia a interação entre os objetos.
- **Objetos:** Representam as instâncias das classes representadas no processo.
- **Linha de vida:** As linhas de vida compõem a dimensão vertical. Uma linha de vida é composta de duas partes, a cabeça e a cauda. A cabeça é representada por um retângulo com dois compartimentos, no compartimento superior a identificação do objeto é exibida e no compartimento inferior (cuja utilização é opcional), aparecem valores para os atributos definidos na classe do objeto. A cauda corresponde a uma linha vertical tracejada.

## Exemplificando diagramas de caso de uso, de classe e de sequência.

<<Andreia, aqui temos um problema de formatação – espaçamento entre as linhas>>

Considere um Software de Gestão Escolar, onde o aluno pode se cadastrar e efetuar sua matrícula via internet ou através da secretaria acadêmica. Para efetivar a matrícula, o aluno deverá efetuar o pagamento na tesouraria. E um aluno matriculado pode solicitar a emissão de seu histórico escolar na secretaria acadêmica, e caso esteja formado, também poderá solicitar a emissão do certificado de conclusão de curso. A secretaria acadêmica por sua vez, também pode cadastrar professores no sistema.

Diante desse cenário podemos exemplificar os três diagramas explicados anteriormente. Na Figura 1, temos o diagrama de Caso de Uso, onde podemos observar os atores Alunos, Secretaria Acadêmica e Tesouraria, e os casos de uso, os quais cada ator interage. E podemos observar os relacionamentos *include* e um com *extend* em dois casos de uso, o *include* pois Efetuar Matrícula inclui Efetuar Pagamento, e o *extend* que estende Emitir Certificado de Conclusão de Curso quando o aluno solicita Emitir Histórico Escolar, caso o aluno esteja formado.

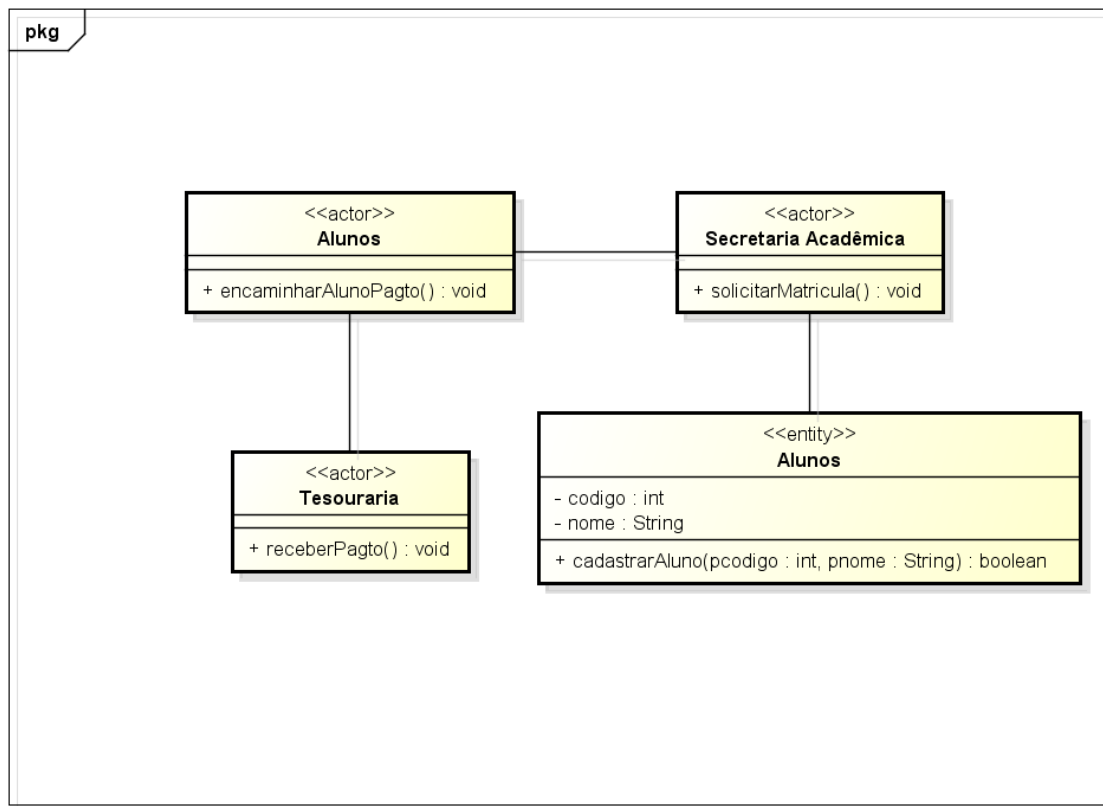


**Figura 1: Diagrama de Caso de Uso.**

Na **Figura 2**, podemos observar o Diagrama de Classe, onde Alunos, Secretaria Acadêmica, Tesouraria e Alunos são as classes. A classe Tesouraria se relaciona com a classe Alunos, que se relaciona com a classe Secretaria Acadêmica, que por sua vez se relaciona com Alunos.

Por meio da **Figura 2** podemos observar também que as classes possuem estereótipos diferentes. Tesouraria, Alunos e Secretaria Acadêmica são classes estereotipadas como atores, enquanto a outra classe, Alunos, é estereotipada como entidade. As classes estereotipadas como entidades irão se tornar uma tabela no banco de dados.

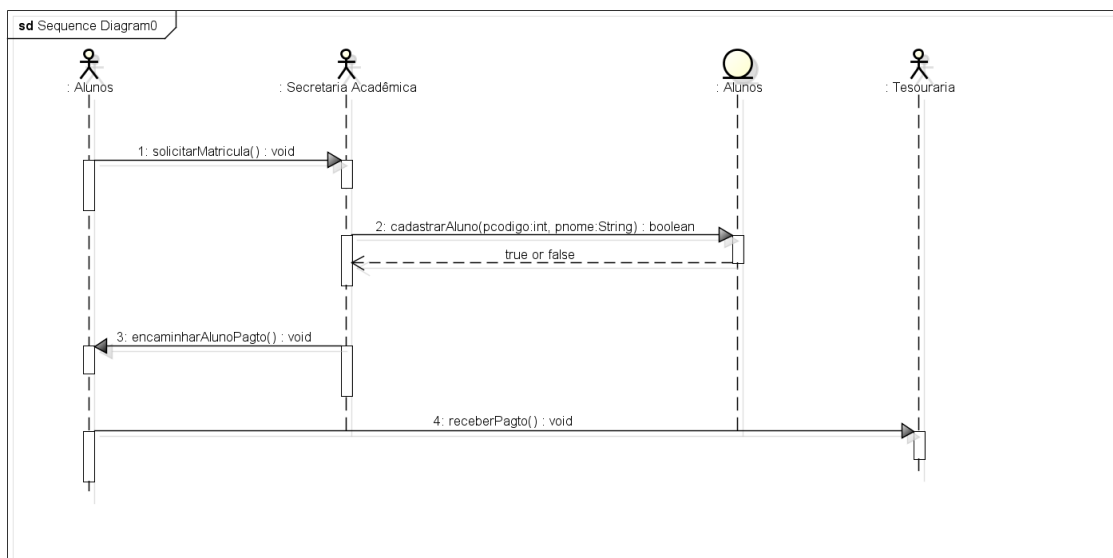
Cada classe possui atributos e métodos, que podem ser observados na **Figura 2**, percebe-se que na classe Alunos os atributos são código e nome, e o método caracteriza-se cadastrarAluno, as classes Tesouraria, Alunos e Secretaria Acadêmica, possuem apenas métodos, que são, respectivamente, receberPagto, encaminharAlunoPagto e solicitarMatricula.



**Figura 2: Diagrama de Classe.**

O Diagrama de Sequência caracteriza-se como o diagrama que representa as ações de determinada cena. A cena é representada pelo caso de uso, ou seja, cada caso de uso corresponde a uma cena.

Cada Diagrama de Sequência estará interligado a um Caso de Uso, na **Figura 3**, temos como exemplo, o Diagrama de Sequência para o Caso de Uso Efetuar Matrícula. Podemos observar os objetos e as mensagens que são trocadas entre eles no decorrer do tempo para a realização de uma operação. Essas mensagens trocadas entre os objetos, são caracterizadas como métodos representados no Diagrama de Classe.



**Figura 3: Diagrama de Sequência.**

Agora com a noção de Diagrama de Caso de Uso, Diagrama de Classe e Diagrama de Sequência mais clara, podemos falar um pouco sobre métricas de software.

## Métricas de software

Para garantir a qualidade de uma aplicação de software existem diversas medidas, dentre elas, a métrica de software. Essa medida auxilia a tomada de decisão, pois através de dados quantitativos, é capaz de informar que aspectos do produto atendem ou não ao padrão de qualidade especificado, além de permitir a avaliação dos benefícios de novos métodos e ferramentas de engenharia de software, o entendimento e aperfeiçoamento do processo de produção e a avaliação do retorno do investimento.

As métricas podem ser categorizadas de maneiras diferentes, tais como métricas diretas, indiretas, orientadas a tamanho e função, de produto e produtividade, de qualidade e técnicas, privadas, públicas e de predição e controle. A **Tabela 2**, apresenta a descrição de cada tipo de métrica citado.

Métricas	Descrição
Diretas	São realizadas em termos de atributos observáveis, como por exemplo, esforço, tamanho e custo.
Indiretas	Podem ser obtidas através de outras métricas, como por exemplo, complexidade, confiabilidade, e facilidade de manutenção.
Orientadas a tamanho	Consideram o tamanho do software produzido, ou seja, a quantidade de linhas de código produzidas.
Orientadas a função	Concentram-se na funcionalidade do software, ou seja, o que é entregue ao cliente, essa métrica consiste em um método para medição de software do ponto de vista do usuário, que determina de forma consistente ao tamanho e complexidade de um software.
De produto	Se ocupam com as características do próprio software, e se dividem em métricas estáticas ou dinâmicas. As métricas estáticas são coletadas por medições feitas das representações do sistema, como projeto, programa ou documentação. As métricas dinâmicas são coletadas por medições feitas de um programa em execução.
De produtividade	Concentram-se na saída do processo de engenharia de software, por exemplo, número de casos de uso e iteração.
De qualidade	Oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente, por exemplo, erros.
Técnicas	Concentram-se nas características do software e não no processo por meio do qual o software foi desenvolvido, por exemplo complexidade lógica, manutenibilidade.
Privadas	Se referem ao escopo da equipe do projeto de software, por exemplo, defeitos para funções importantes do software, erros encontrados durante revisões e técnicas formais.
Públicas	Geralmente assimilam informações que anteriormente eram privadas de uma equipe, por exemplo, proporções de defeitos de projeto, esforço, tempo transcorrido e dados relacionados, são

**Tabela 2: Tipos de métricas**

A utilização das métricas de software se faz necessária para possibilitar melhorias e resultados mais satisfatórios do software, mais segurança para os gerentes de projeto. Sua utilização também é uma maneira de eliminar os obstáculos, corrigir erros e falhas, antes mesmo do produto ser entregue ao cliente.

Alguns exemplos de métricas de software existentes são LOC (número de linhas de código), pontos por função e pontos por caso de uso. Neste artigo o foco será a métrica de pontos por caso de uso.

[A partir daqui inicia a parte **PRÁTICA** do artigo – Mantenha a Tag abaixo, será usada para contagem automática usando macros]

[CHECKPOINT]

## Pontos por caso de uso

Para calcular a complexidade de um software utilizando pontos por caso de uso é necessário calcular a complexidade dos atores e a complexidade dos casos de uso.

### Complexidade dos atores do software

A complexidade dos atores é calculada classificando os atores envolvidos em cada caso de uso, de forma a obter um somatório de pontos não ajustados. A classificação de atores é dada pela **Tabela 3**, e o cálculo é realizado analisando todos os atores dos diagramas de caso de uso e classificando-os, o resultado total é o somatório da complexidade de cada ator.

Ator	Interface	Peso
Simples	Interface de programa (API)	1
Médio	Protocolo (Ex.:TCP/IP) ou interface em modo texto	2
Complexo	Interface gráfica	3

**Tabela 3: Classificação da complexidade dos atores**

Na Tabela 3, pode-se observar três tipos de interações do ator com o sistema, por meio de interface de programa, protocolo ou interface em modo texto e interface gráfica.

Um ator é considerado interface de programa, ou API quando permite a utilização de características do software menos evidentes ao usuário final, é composta por uma série de funções acessíveis somente por programação, ou seja, o ator é um programa, um exemplo de API será explicado posteriormente na forma como o *plugin* foi desenvolvido.

Um ator pode ser caracterizado como um protocolo, ou seja, um meio comum para objetos não relacionados se comunicarem uns com os outros, ou interface em modo texto (CLI), quando o usuário interage com um sistema através de linhas de comando, ou seja, possibilita ao usuário interagir com o sistema digitando comandos para que o computador realize tarefas específicas.

E por fim, um ator pode ser considerado como uma interface gráfica, essa classificação determina o ator como uma janela gráfica para que o usuário final interaja diretamente com o sistema, sem a necessidade de utilizar nenhuma linguagem de programação.

Um exemplo prático para o cálculo da complexidade dos atores (ca) utilizando como base o diagrama de caso de uso representado na **Figura 1**, onde os três atores são complexos, ou seja, acessam o software por meio de uma interface gráfica, portanto, possuem peso igual a 3, pode ser visto abaixo,

$$\begin{aligned} ca &= 3 + 3 + 3 \\ ca &= 9 \end{aligned}$$

## Complexidade dos casos de uso do software

A complexidade dos casos de uso é calculada classificando-os em simples, médio ou complexo, conforme a **Tabela 4**, esse cálculo é feito utilizando como base diagramas de sequência, e o resultado é obtido pelo somatório dos pesos de cada tipo de caso de uso.

Caso de Uso	Descrição	Peso
Simples	$\leq 3$ transações ou $< 5$ classes de análise	5
Médio	4-7 transações ou 5 a 10 classes de análise	10
Complexo	$> 7$ transações ou $> 10$ classes de análise	15

**Tabela 4: Classificação dos casos de uso**

Analisando a **Tabela 4**, os parâmetros para análise em cada diagrama podem ser dois, as transações ou as classes de análise.

Os diagramas de sequência são diagramas de interação enfatizando o tempo de sequência, ele mostra também objetos participando em interações de acordo com suas linhas de vida e as mensagens trocadas, utilizando como exemplo o diagrama da **Figura 3**, na qual os objetos são Alunos, Secretaria Acadêmica e Tesouraria e a interface Alunos, e as interações são os métodos solicitarMatricula, cadastrarAluno, encaminharAlunoPagto e receberPagto, essas interações são o mesmo que as transações na classificação dos casos de uso, ou seja, nesse caso o diagrama de sequência possui 4 transações, portanto a complexidade do caso de uso seria média, peso 10.

Pode-se também realizar esse cálculo através das classes de análise, onde é necessário contar os objetos (que são caracterizados como instâncias das classes) presentes no diagrama de sequência, que neste caso seriam Alunos, Secretaria Acadêmica e Tesouraria e a interface Alunos, ou seja, 4 classes de análise, portanto, complexidade média, peso 10.

Concluído os cálculos das duas etapas explicadas, pode-se obter o valor dos pontos por caso de uso (pcu) realizando o somatório da complexidade dos atores e da complexidade do caso de uso, que no exemplo dado seria,

$$\begin{aligned} pcu &= 9 + 10 \\ pcu &= 19 \end{aligned}$$

## O Astah

Um dos maiores desafios para a construção de um software está relacionado ao correto entendimento das necessidades do cliente, para que se possa então prover uma solução computacional capaz de atender a estas necessidades. Para prover essa solução deve-se fazer a modelagem do software em questão. Existem diversas ferramentas de modelagem de software entre elas o Astah.

O Astah é uma ferramenta para modelagem UML (Unified Modeling Language – Linguagem de Modelagem Unificada), criado pela empresa japonesa Change Vision e desenvolvido na plataforma Java.



No site da ferramenta (veja a sessão links), pode-se perceber que o Astah possui várias versões para download, dentre elas o Astah Community e o Astah Professional, a versão Community é uma versão *free*, porém, apresenta algumas limitações, para o desenvolvimento do *plugin* é necessário utilizar a versão Professional, que apesar de ser uma versão paga, é disponibilizada 30 dias para teste e é possível obter uma licença para fins acadêmicos.

No site também, encontra-se disponível para download vários *plug-ins*, porém, caso o usuário não encontre um que satisfaça as suas necessidades, ele poderá apresentar um pedido no site, ou desenvolver o seu próprio *plugin*.

Nesse artigo, a partir da teoria explicada anteriormente, será apresentado um *plugin*, desenvolvido pela autora deste trabalho, para o cálculo dos pontos por caso de uso. Para o desenvolvimento do plugin foram utilizados tutoriais e a documentação da API da ferramenta, estes artefatos podem ser obtidos no site da ferramenta astah (veja a seção links). Com os tutoriais foi possível configurar a IDE Eclipse e o Astah SDK disponibilizado pela *Change Vision*. Após tudo configurado corretamente foi possível iniciar o desenvolvimento do *plugin*.

Conhecendo todos os conceitos de pontos por caso de uso, e com a documentação da API em mãos pode-se iniciar o desenvolvimento. Através da API foi possível descobrir como acessar cada elemento necessário para o cálculo, dos pontos por caso de uso: os atores envolvidos em cada caso de uso e as interações entre os objetos nos diagramas de sequência.

## Desenvolvimento do plugin

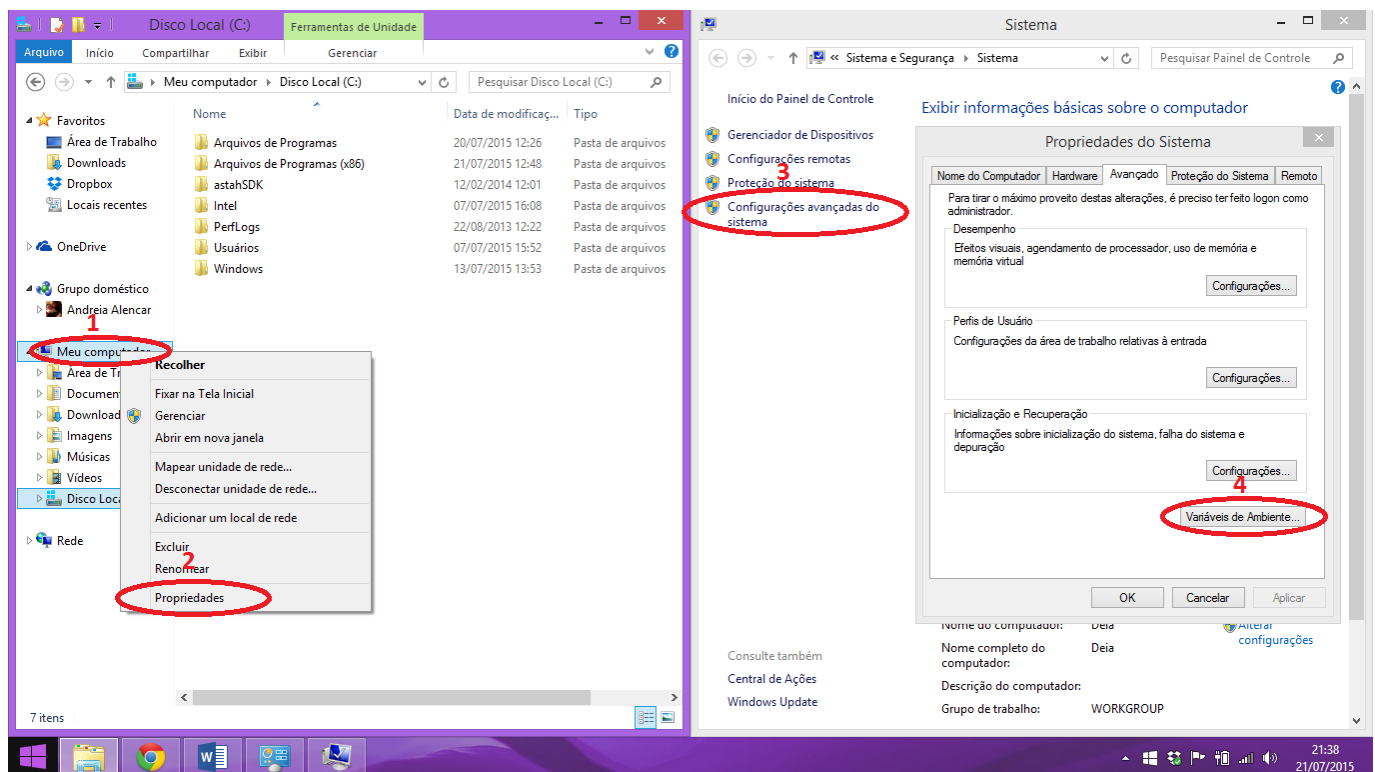
Nesta seção iremos apresentar noções básicas para o desenvolvimento de um *plugin*, que tem como finalidade apresentar uma mensagem de “hello world” na tela para o usuário, pois para apresentar o desenvolvimento completo do *plugin* proposto, o artigo se tornaria demasiadamente complexo e amplo. Ao conseguir desenvolver o “hello world” o leitor será capaz de desenvolver qualquer tipo de *plugin*, desde que tenha conhecimento na linguagem de programação Java.

Os pré-requisitos para iniciar o desenvolvimento de um *plugin* é ter instalado na máquina a IDE Eclipse, o kit de desenvolvimento de software (SDK) do astah e o Java JDK.

Para desenvolver um *plugin* no Astah, é necessário configurar devidamente o ambiente. Para isso, deve-se acessar o link para download do SDK (veja a sessão links) e efetuar o download do Astah Plug-in SDK.

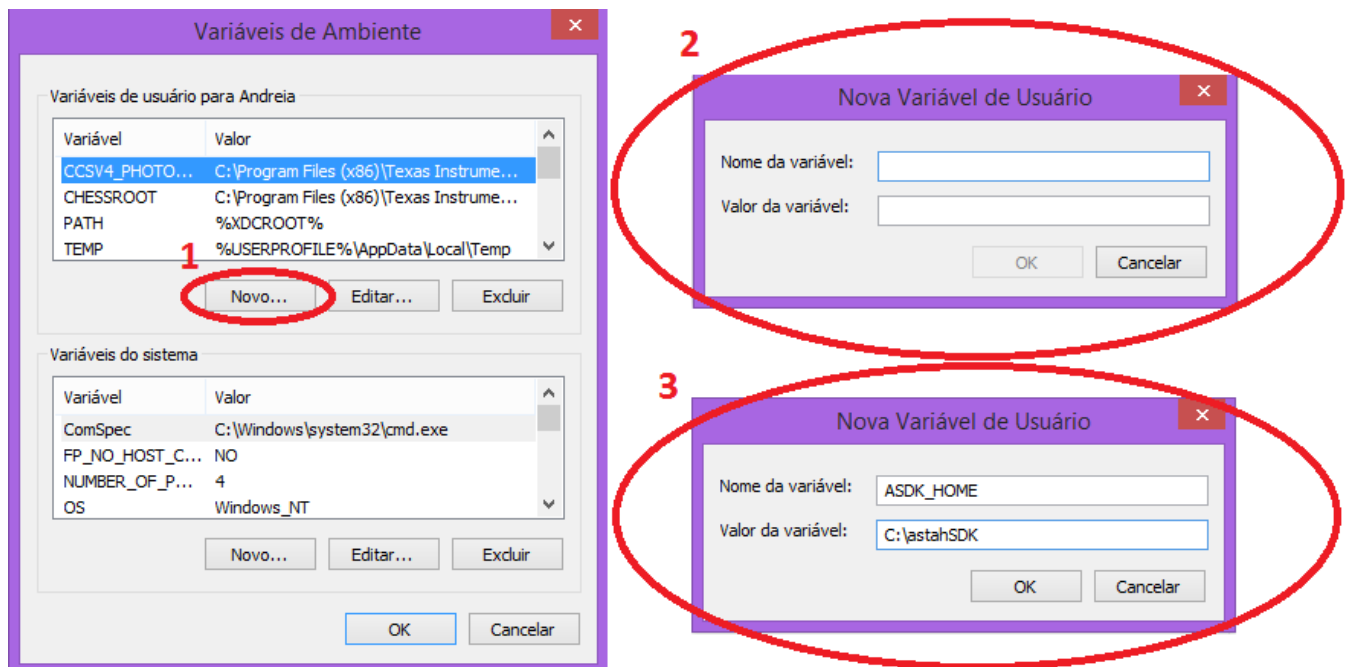
O Astah Plugin SDK estará em formato zip, portanto, deverá ser extraído, e então teremos uma pasta chamada “astah-plugin-SDK-1.2”, que deverá ser renomeada para “astahSDK”, por questão de facilidade, e colocada na pasta raiz do computador.

Em seguida, deve-se configurar as variáveis de ambiente do Windows, a **Figura 4**, apresenta os passos para acessar essas variáveis, primeiramente, deve-se clicar com o botão direito do mouse sobre “Meu computador” (1), em seguida Propriedades (2) >> Configurações avançadas do sistema (3) >> Variáveis de Ambiente (4).



**Figura 4: Passo a passo para acessar as variáveis de ambiente do Windows**

Após seguir os passos da **Figura 4**, será encontrada a janela representada pela **Figura 5**, onde serão configuradas as variáveis de ambiente para possibilitar o uso do astahSDK para o desenvolvimento do *plugin*.

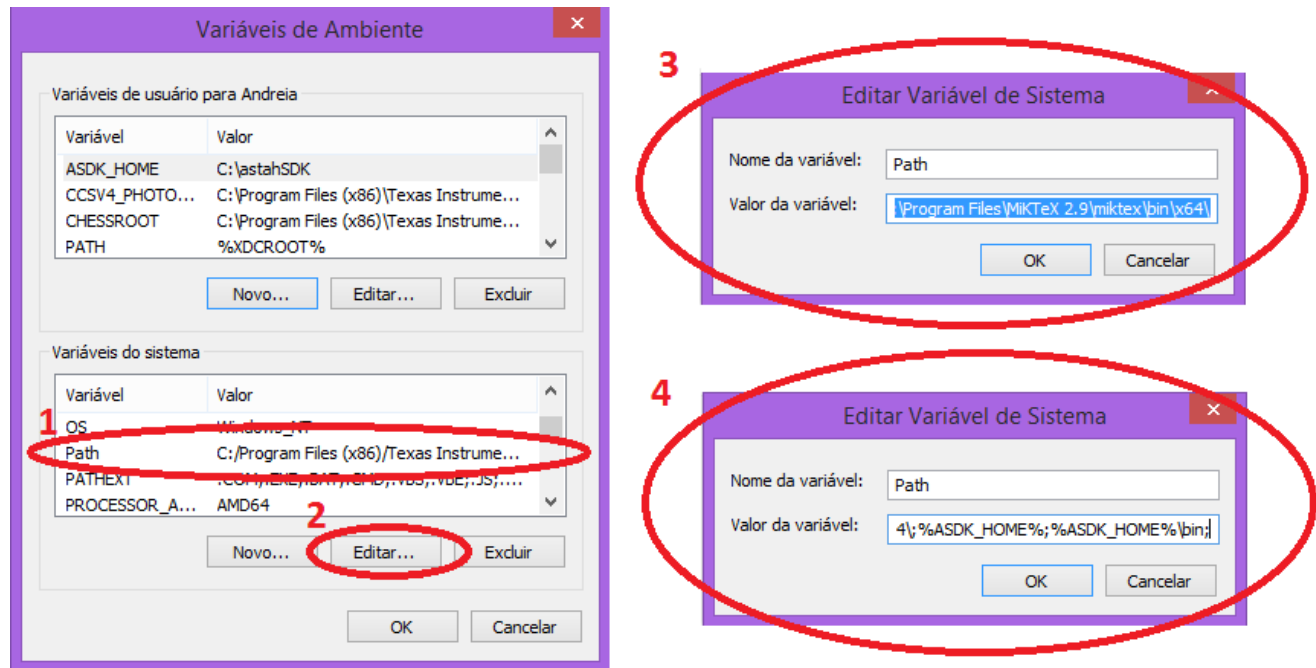


**Figura 5: Criando nova Variável de Ambiente**

O primeiro passo para configurar as variáveis de ambiente, será criando uma nova variável chamada “ASDK\_HOME”, para isso deve-se clicar no botão “Novo...” (1), onde será aberta a janela

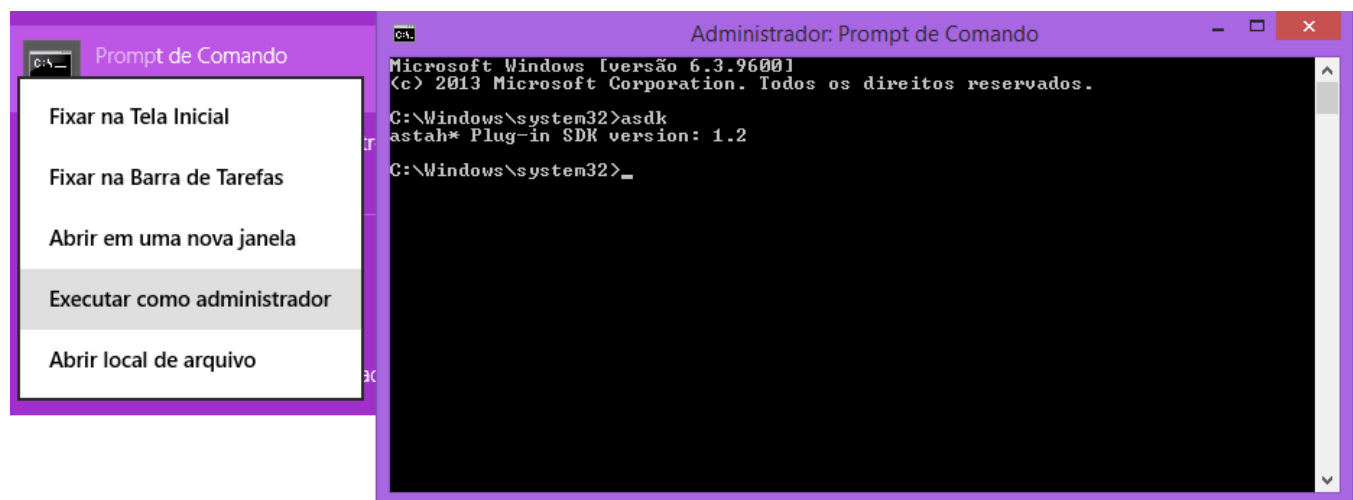
“Nova Variável de Usuário” (2), no campo “Nome da variável”, deve-se digitar “ASDK\_HOME”, e no campo “Valor da variável”, o caminho para a pasta “astahSDK”, como pode ser observada em (3) na **Figura 5**.

Após criar a variável “ASDK\_HOME”, deve-se adicioná-la na variável Path do Windows, da seguinte forma: “%ASDK\_HOME%;%ASDK\_HOME%\bin;”, para isso, deve-se selecionar a variável Path, como demonstrado em (1), na **Figura 6**, em seguida, clicar no botão “Editar...” (2), e adicionar “%ASDK\_HOME%;%ASDK\_HOME%\bin;”, ao final do campo Valor da variável, como pode ser observado em (4), na **Figura 6**.



**Figura 6:** Adicionando a variável criada ao Path do Windows

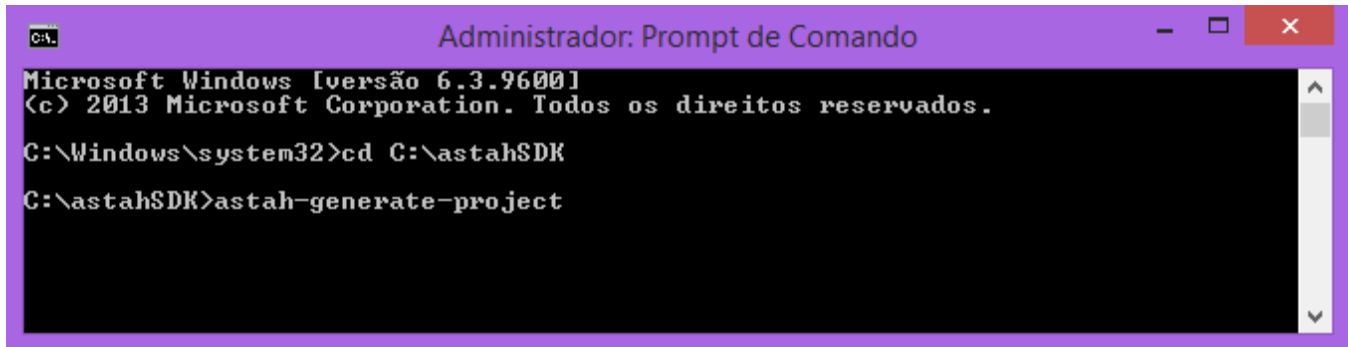
Para testar se a configuração foi realizada corretamente, é necessário digitar o comando “asdk” no *prompt de comando* do Windows, executado como administrador, se a configuração estiver correta, aparecerá “astah\* Plug-in SDK version: 1.2” na tela, como pode ser observado na **Figura 7**.



**Figura 7:** Teste da configuração das Variáveis de Ambiente

Após ter o ambiente devidamente configurado, o *plugin* já pode ser desenvolvido, um exemplo bem simples para começar a trabalhar é o *HelloWorld*.

Para iniciar é necessário abrir o *prompt de comando* do Windows, também como administrador, e acessar a pasta “astahSDK”, onde foi instalado o *SDK*, com o comando *cd*, dentro da pasta deve ser digitado o comando “astah-generate-project”, para criar um novo projeto. Como mostra a **Figura 8**.



```
Administrador: Prompt de Comando
Microsoft Windows [versão 6.3.9600]
(c) 2013 Microsoft Corporation. Todos os direitos reservados.

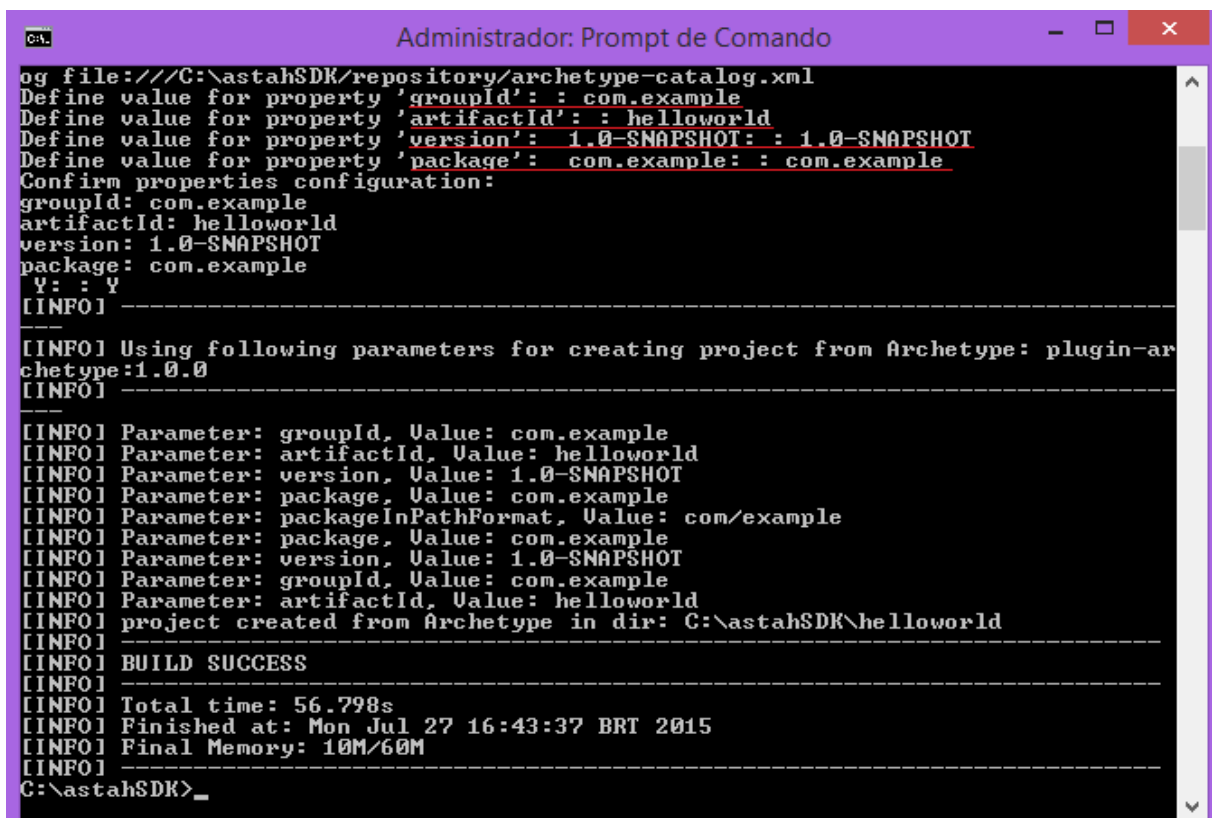
C:\Windows\system32>cd C:\astahSDK
C:\astahSDK>astah-generate-project
```

Figura 8: Comando para criar novo projeto

Após digitar o comando, algumas informações serão solicitadas, e devem ser preenchidas da seguinte forma:

- **groupId:** com.example
- **artifactId:** helloworld
- **version:** 1.0-SNAPSHOT
- **package:** com.example,

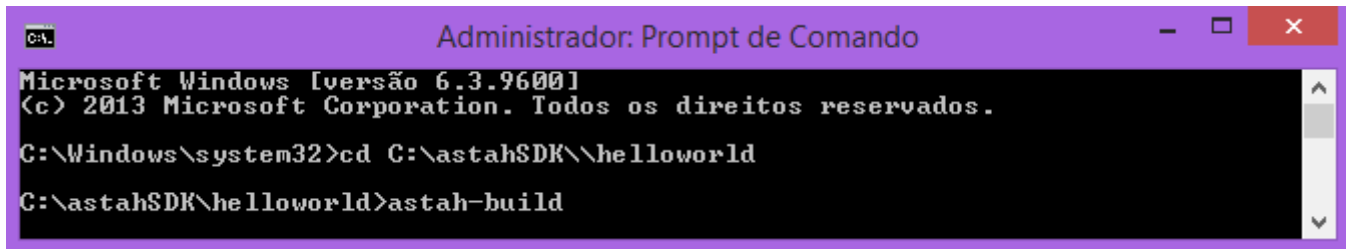
após preencher essas informações, será solicitada a confirmação, para isso basta digitar o caractere “y”. E então o projeto estará construído, como mostra a **Figura 9**.



```
Administrador: Prompt de Comando
og file:///C:/astahSDK/repository/archetype-catalog.xml
Define value for property 'groupId': : com.example
Define value for property 'artifactId': : helloworld
Define value for property 'version': : 1.0-SNAPSHOT
Define value for property 'package': : com.example
Confirm properties configuration:
groupId: com.example
artifactId: helloworld
version: 1.0-SNAPSHOT
package: com.example
Y: : Y
[INFO] -----
[INFO] Using following parameters for creating project from Archetype: plugin-ar
chetype:1.0.0
[INFO] -----
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: helloworld
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: packageInPathFormat, Value: com/example
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: helloworld
[INFO] project created from Archetype in dir: C:\astahSDK\helloworld
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 56.798s
[INFO] Finished at: Mon Jul 27 16:43:37 BRT 2015
[INFO] Final Memory: 10M/60M
[INFO] -----
C:\astahSDK>
```

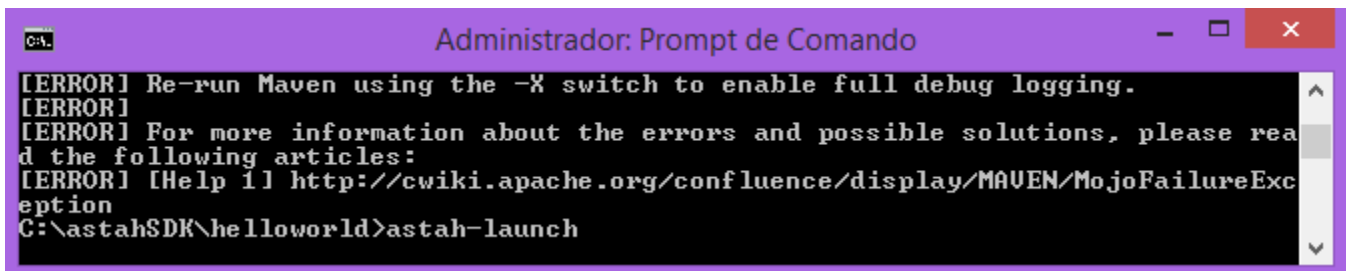
Figura 9: Informações do novo projeto

O próximo passo, é acessar a pasta do projeto criado, ou seja, do projeto chamado “helloworld”, ainda através do *prompt de comando* como administrador, dentro da pasta deve ser dado o comando “astah-build”, como mostra a **Figura 10**, para compilar o projeto. A primeira vez em que esse comando é executado, seu processamento será mais demorado que nas posteriores.



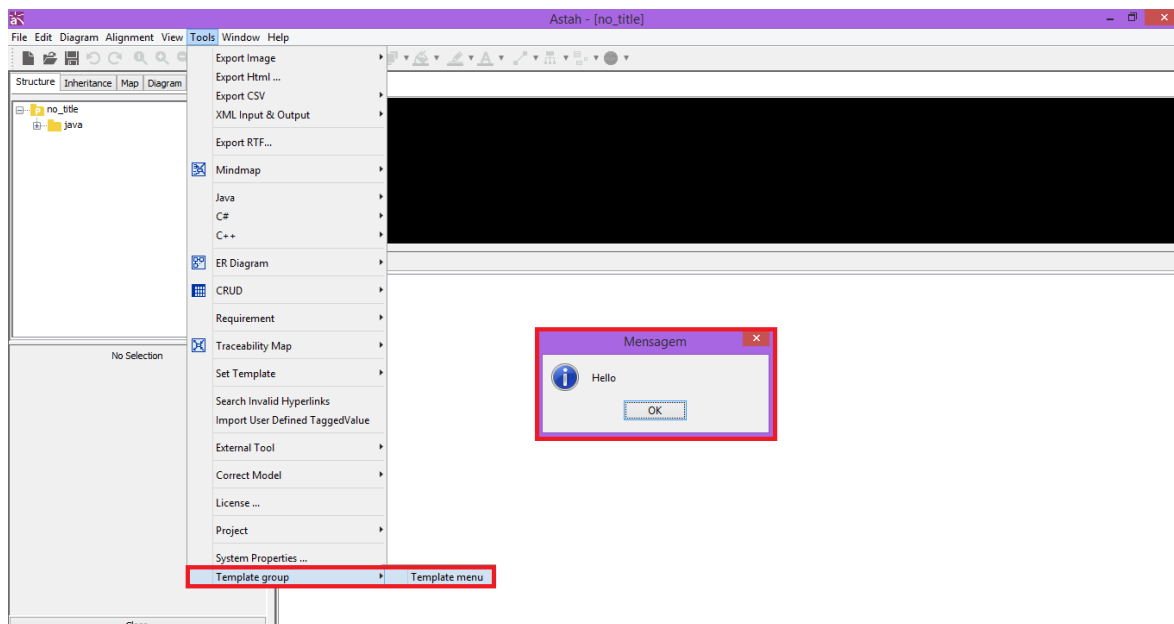
**Figura 10:** Compilando o projeto criado

Após compilado, deve-se dar o comando “astah-launch”, como mostra a **Figura 11**, esse comando irá abrir a ferramenta astah com o plugin que acabou de ser compilado.



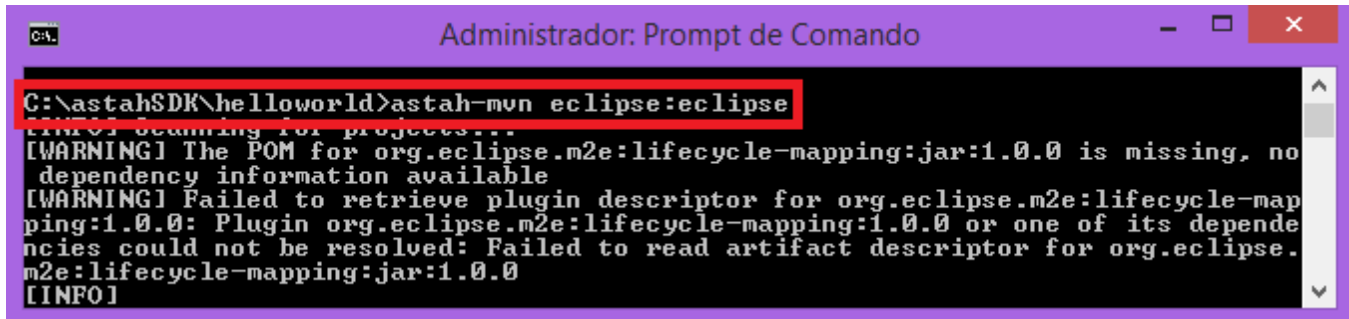
**Figura 11:** Comando para executar a ferramenta Astah, com o plugin em desenvolvimento configurado

Para verificar a funcionalidade do plugin, deve-se criar um novo projeto astah ou abrir um projeto já existente, e acessar na aba “Tool” o menu “Template menu”, que se encontra dentro do menu “Template group” (o nome dos menus podem ser futuramente alterados), e uma janela com a palavra “Hello”, será aberta, como mostra a **Figura 12**, ou seja, o plugin está funcionando.



**Figura 12:** Testando o *plugin* criado

Para acessar o código fonte do plugin, deve-se ter a IDE Eclipse instalada, e dar o comando “astah-mvn eclipse:eclipse”, no prompt de comando, ainda dentro da pasta do projeto, como mostra a **Figura 13**, em seguida, deve-se abrir a IDE e importar o projeto criado, o caminho para isso é File >> Import >> General >> Existing Projects into Workspace, encontrar a pasta do projeto criado e então importá-lo, como ilustra a **Figura 14**.



```
C:\astahSDK\helloworld>astah-mvn eclipse:eclipse
[INFO] Scanning for projects...
[WARNING] The POM for org.eclipse.m2e:lifecycle-mapping:jar:1.0.0 is missing, no
dependency information available
[WARNING] Failed to retrieve plugin descriptor for org.eclipse.m2e:lifecycle-map
ping:1.0.0: Plugin org.eclipse.m2e:lifecycle-mapping:1.0.0 or one of its depende
ncies could not be resolved: Failed to read artifact descriptor for org.eclipse.
m2e:lifecycle-mapping:jar:1.0.0
[INFO]
```

Figura 13: Comando para acessar código fonte através do Eclipse

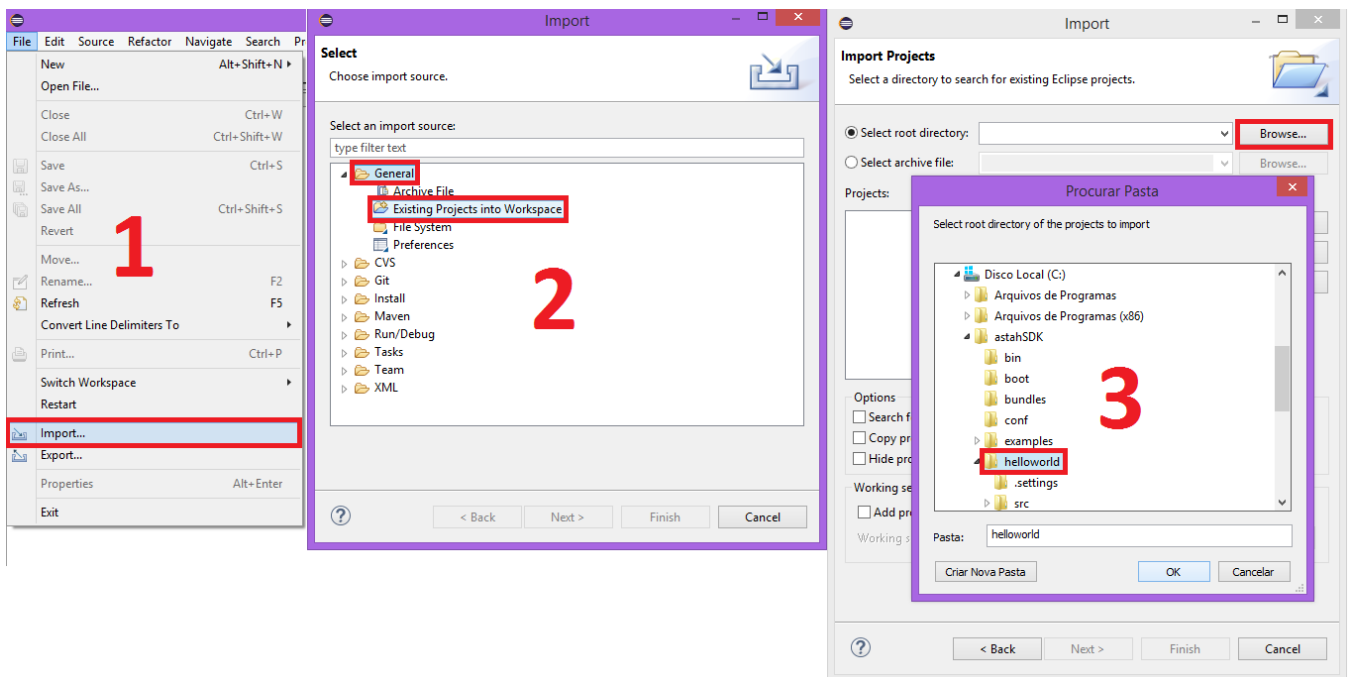


Figura 14: Passo a passo para importar projeto no Eclipse

Dentro do pacote com.example do projeto, que se encontra em “src/main/java”, a classe TemplateAction, é a classe em que se deve trabalhar para desenvolver as funcionalidades desejadas para o *plugin*. Para teste pode-se alterar a linha “JOptionPane.showMessageDialog (window.getParent(),”Hello”);” para “JOptionPane.showMessageDialog (window.getParent(),”Hello World”);”, como exemplificado na **Figura 15**, e realizar novamente o processo para compilar o *plugin* e iniciar a ferramenta com ele, e agora uma janela com a palavra “Hello World”, deverá ser aberta, quando acessar o *plugin*.

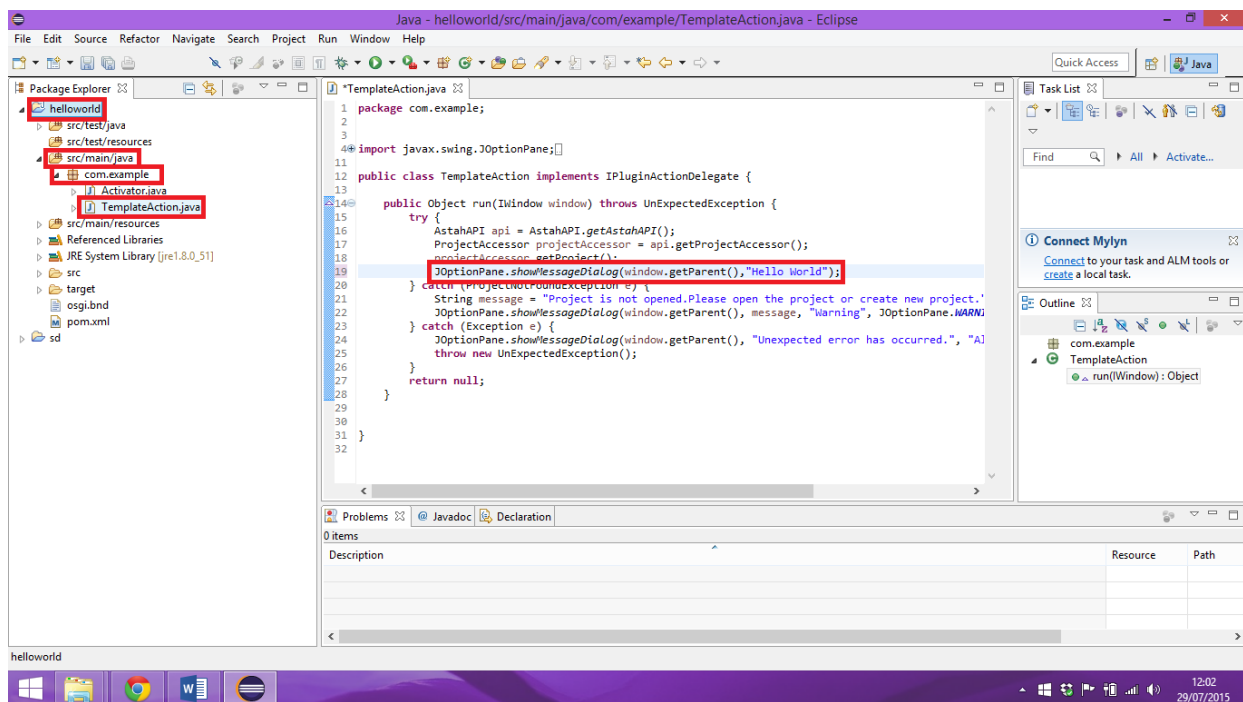


Figura 15: Testando código do *plugin*

Até aqui tivemos noções para o desenvolvimento de *plug-ins* para a ferramenta astah, e com os tutoriais encontrados no site (ver seção links), e com dicas em fóruns e blogs é possível desenvolver *plug-ins* com funcionalidades diversas para facilitar no desenvolvimento de projetos de software, utilizando a ferramenta astah.

Agora vamos falar sobre o *plug-in* que foi desenvolvido, como instalá-lo na ferramenta e como utilizá-lo em seu projeto.

## Instalação do plugin desenvolvido

Para utilizar o *plugin*, apresentado neste artigo, na ferramenta astah profissional, é necessário realizar o download (veja sessão links), e diferente do que foi explicado anteriormente para testar se o *plugin* está funcionando corretamente, a instalação do mesmo deve ser realizada, assim toda vez que abrirmos a ferramenta, o *plugin* estará instalado e pronto para ser utilizado.

Para instalação do *plugin* existem dois métodos diferentes, o primeiro é colocando o arquivo “usecasepoints-1.0.jar” na pasta “plugin” que se encontra dentro da pasta de instalação da ferramenta, no caminho “C:\Program Files\astah-professional\plugins”, como pode ser visto na **Figura 16**.

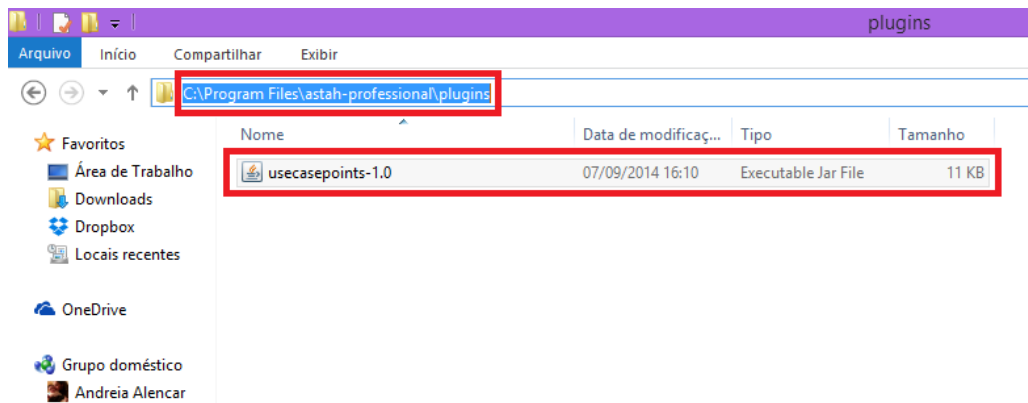


Figura 16: Primeiro método de instalação do *plugin*

O segundo método é utilizando o menu *plugin list*, que se encontra na aba *help* da ferramenta, o caminho pode ser observado na **Figura 17**. Após clicar no menu *plugin list*, o próximo passo pode ser observado na **Figura 18**, só clicar em *Install*, seleccionar o *plugin* e clicar em *Abrir*, o *plugin* estará instalado.

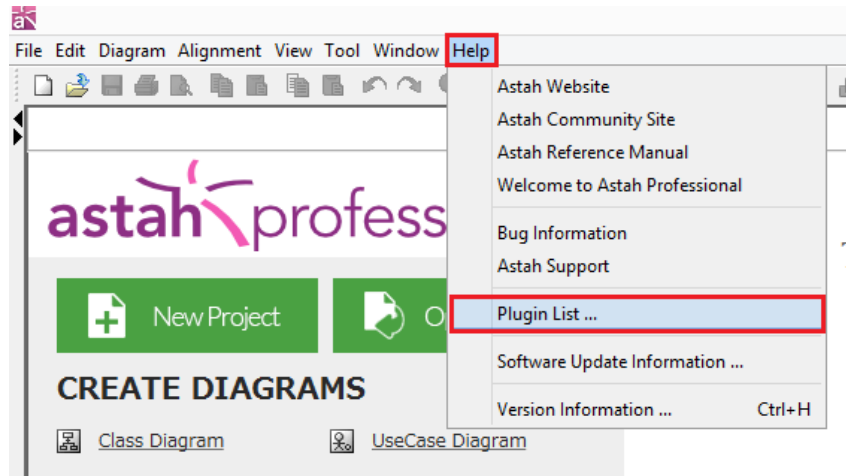


Figura 17: Segundo método de instalação do *plugin* - parte 01

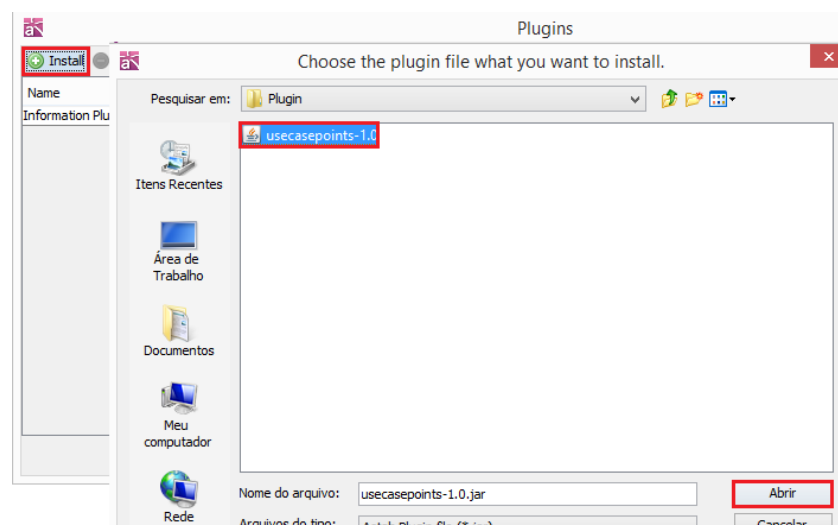


Figura 18: Segundo método de instalação do *plugin* - parte 02



## Como utilizar o plugin

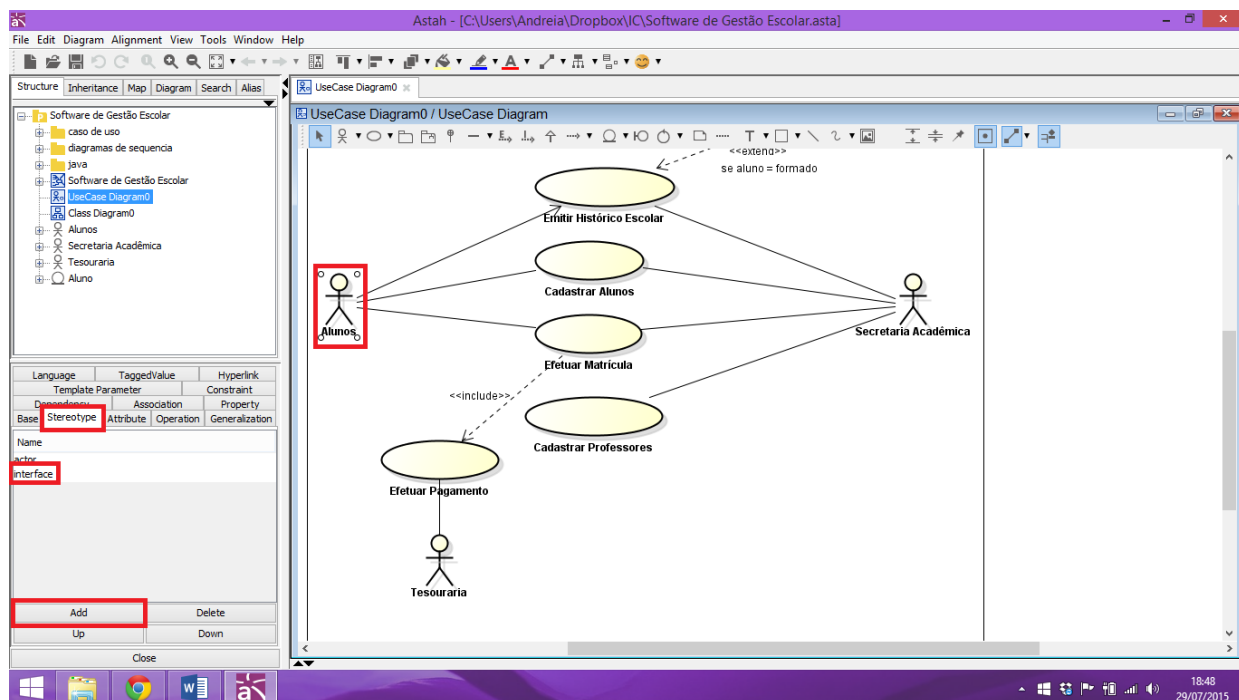
Após instalado, o *plugin* desenvolvido está pronto para calcular os pontos por caso de uso do seu projeto astah, mas algumas regras devem ser seguidas. O projeto deverá conter diagrama de caso de uso e diagrama de sequência.

No diagrama de caso de uso os atores devem ser classificados corretamente, para que o cálculo realizado tenha o resultado esperado. Como visto na teoria os atores podem ser classificados em simples, médio e complexo. No projeto astah para classificá-los devem ser usadas as nomenclaturas de acordo com a Tabela 3.

Ator	Interface	No Astah
Simple	Interface de programa (API)	api
Médio	Protocolo (Ex.:TCP/IP) ou interface em modo texto	tcp
Complexo	Interface gráfica	interface

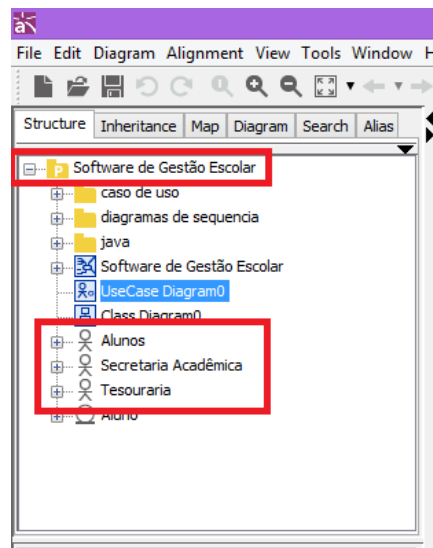
**Tabela 5: Classificando atores no astah**

Na ferramenta essa classificação deve ser feita com o ator selecionado e na aba *Stereotype*, no canto inferior esquerdo adicionar a classificação do ator, como pode ser visto na **Figura 6**.



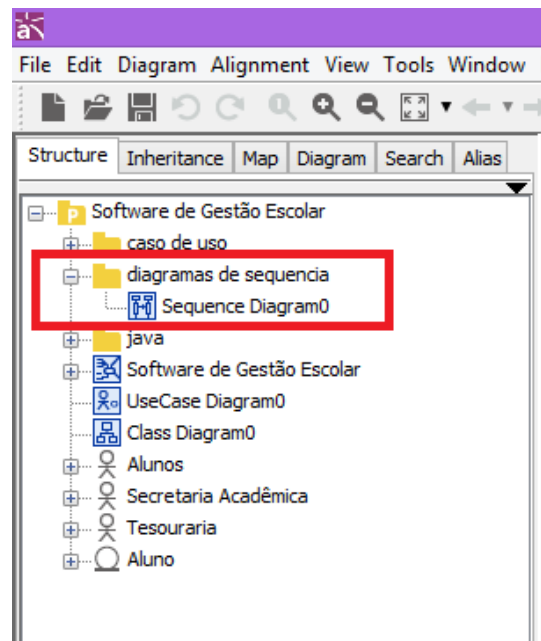
**Figura 19: Classificação de atores no astah**

Uma outra condição para que o *plugin* funcione como desejado, os atores não poderão estar dentro de nenhum pacote, deverão estar na pasta raiz do projeto, como mostrado na Figura 20.



**Figura 20: Condição para classificar atores no astah**

Após classificar os atores do diagrama de caso de uso corretamente, como dito anteriormente, deve-se ter também um diagrama de sequência e este deve estar dentro de um pacote nomeado como “diagrama de sequencia”, como pode ser observado na **Figura 21**.



**Figura 21: Localização do diagrama de sequencia no projeto astah**

Com o projeto pronto, contendo os dois diagramas necessários, e devidamente estruturados, os pontos por caso de uso podem ser calculados automaticamente, para isso o usuário deverá acessar a aba “Tool”, selecionar “Pontos por caso de uso” e em seguida “calcular”, como pode ser visto na **Figura 22**.

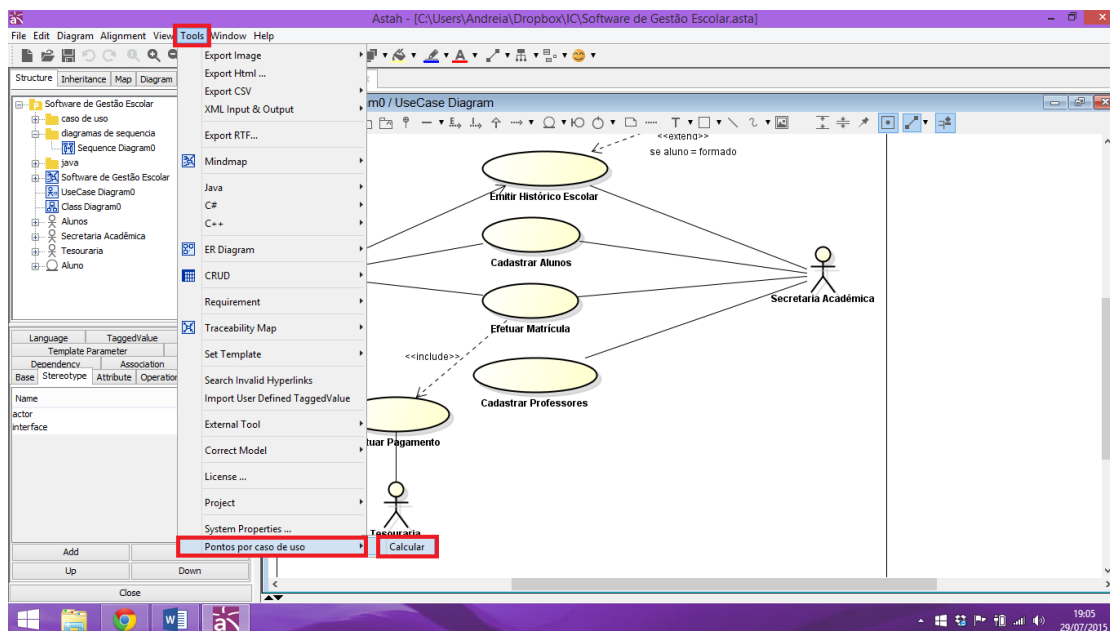


Figura 22: Calculando os pontos por caso de uso através do plugin no astah

## Exemplo de utilização do plugin

Para demonstrar o funcionamento do *plugin* desenvolvido, será utilizado o cenário de um Software de Gestão Escolar apresentado como exemplo na parte teórica deste artigo.

Para isso vamos utilizar o Diagrama de Caso de Uso, exemplificado na **Figura 1** e o Diagrama de Sequência, exemplificado na **Figura 3**, lembrando que em um projeto normalmente existem mais que um Diagrama de Sequência, pois cada Diagrama de Sequência está interligado a um Caso de Uso, e no exemplo temos mais de um caso de uso, porém, aqui iremos utilizar apenas um Diagrama de Sequência, para ilustrar o funcionamento do *plugin*.

Para começar o desenvolvimento, deve-se abrir a ferramenta astah, acessar a aba File, e clicar em New, para criar um novo projeto astah, como pode ser visto na **Figura 23**. Em seguida iremos criar o Diagrama de Caso de Uso, para isso deve-se acessar a aba Diagram e clicar em UseCaseDiagram, como ilustrado na **Figura 24**, e criar o diagrama representado na **Figura 1**. Após criado o diagrama os atores devem ser configurados de acordo com a Tabela 4, como demonstrado na **Figura 25**, no caso do nosso exemplo os três atores, Alunos, Secretaria Acadêmica e Tesouraria são interfaces gráficas, portanto, cada um terá peso equivalente a 3, e devem ter *Stereotype* “interface”, lembrando que os atores não podem estar dentro de nenhum pacote.

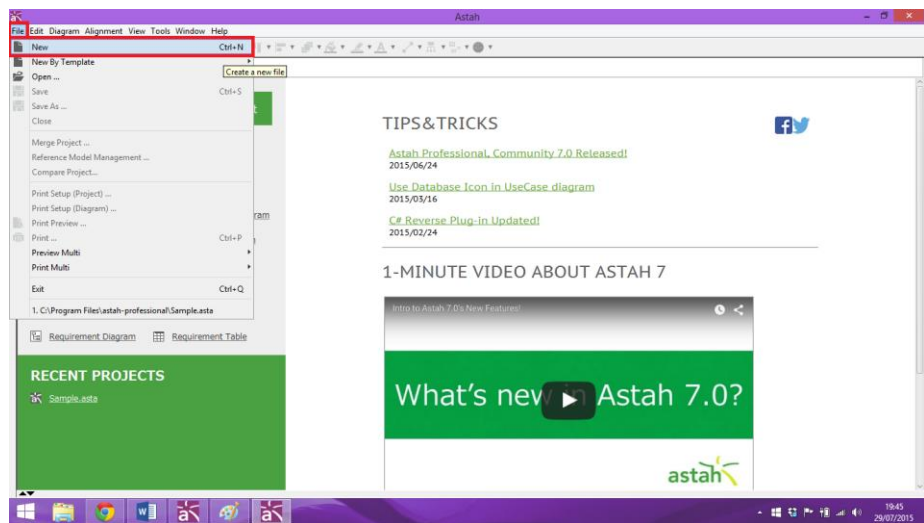


Figura 23: Criando um novo projeto no astah

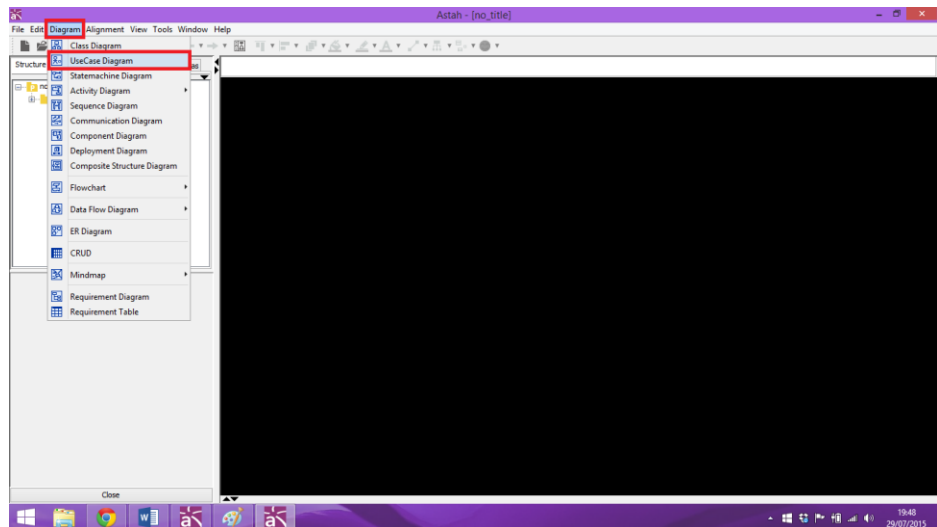


Figura 24: Criando um novo diagrama de caso de uso

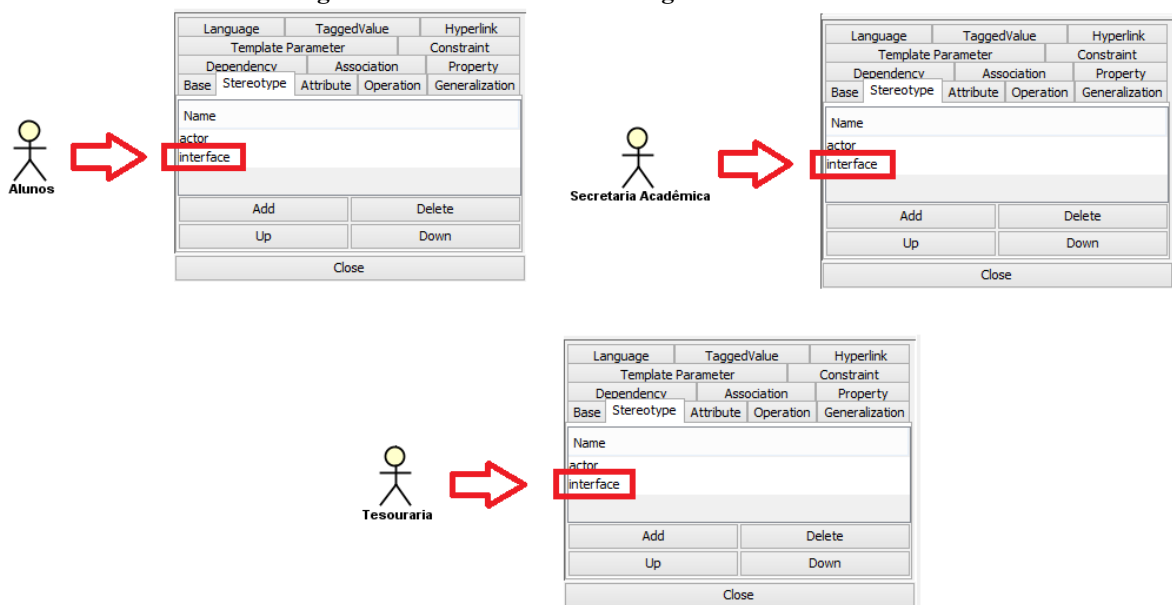
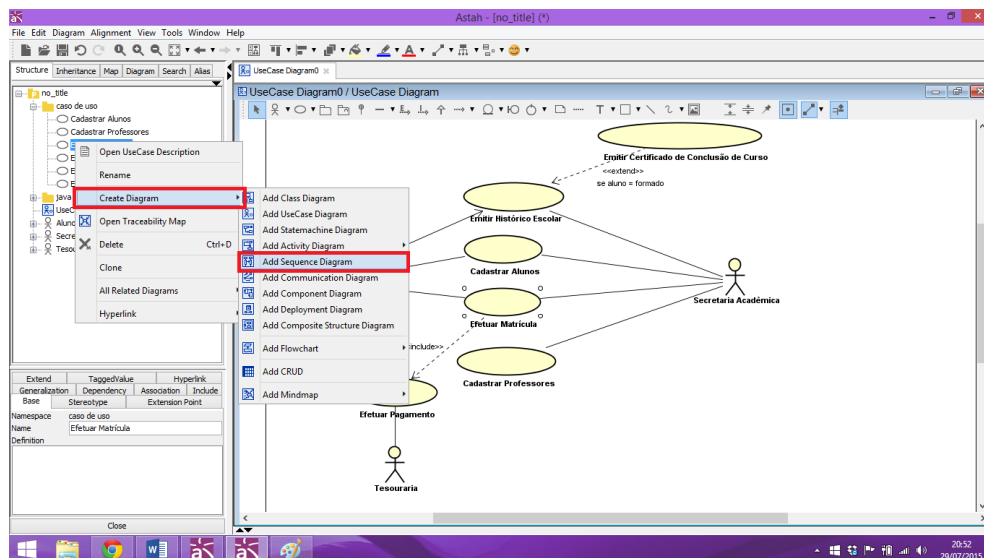


Figura 25: Classificando esteriótipos dos atores no astah

Com o Diagrama de Caso de Uso pronto e os atores devidamente configurados, devemos criar o Diagrama de Sequência. Porém, como as transições do Diagrama de Sequência são os métodos das classes que estão definidas no Diagrama de Classe, é aconselhável criar primeiramente o Diagrama de Classe, para isso deve-se acessar a aba Diagram e clicar em Class Diagram, seguindo o mesmo exemplo para criação do diagrama de caso de uso, e criar o diagrama exemplificado na **Figura 2**.

Após finalizar o Diagrama de Classe, criar o Diagrama de Sequência, no exemplo para o caso de uso Efetuar Matrícula, portanto o caminho é clicar com o botão direito do mouse no caso de uso, acessar Create Diagram e Add Sequence Diagram, como demonstrado na **Figura 26**, e desenvolver o diagrama exemplificado na **Figura 3**, lembrando que as Message entre as classes são os métodos do Diagrama de Classe.



**Figura 26: Criando diagrama de sequencia para o caso de uso Efetuar Matrícula**

Com o Diagrama de Sequência criado, deve-se criar um novo pacote no projeto, como mostra a **Figura 27**, e nomeá-lo como “diagramas de sequencia”, onde deve ser colocado todos os Diagramas de Sequência desenvolvidos, que no caso do nosso exemplo é apenas o Diagrama de Sequência do caso de uso Efetuar Matrícula, a janela obtida será a representada na **Figura 28**.

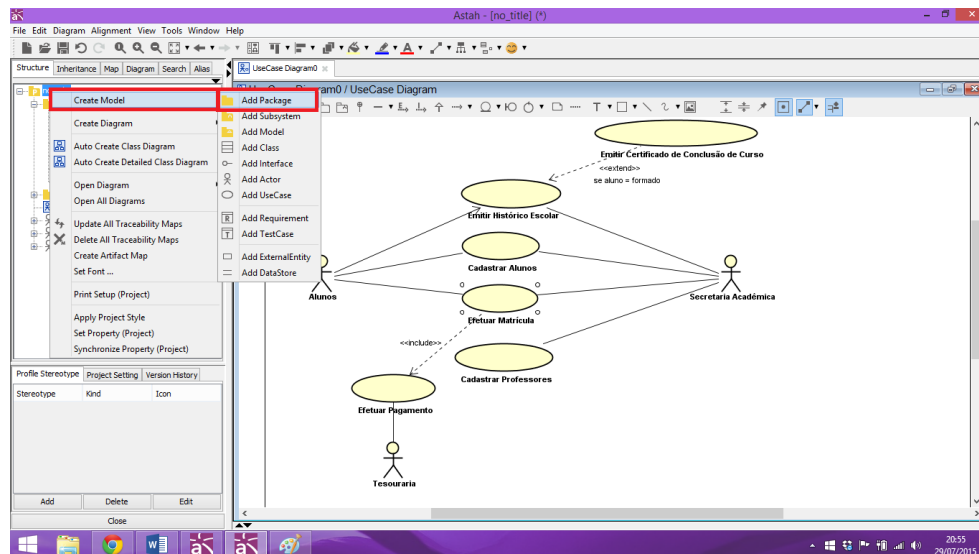


Figura 27: Criando um pacote no projeto astah

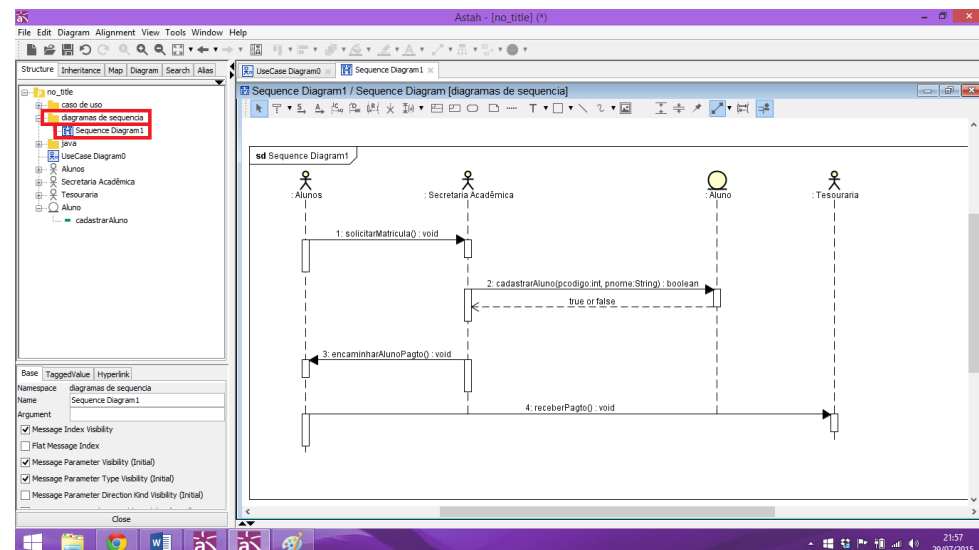


Figura 28: Localização dos diagramas de sequencia no astah para utilização do *plugin*

Agora que temos o Diagrama de Caso de Uso devidamente configurado, e o Diagrama de Sequência, já podemos utilizar o *plugin*, conforme mostra a **Figura 22**.

De acordo com a teoria que foi explicada no tópico sobre Pontos por Caso de Uso, devemos obter como resultado dos pontos por caso de uso o valor 19, pois temos três atores com peso igual a 3 e um Diagrama de Sequência com quatro transações, ou seja, peso igual a 10. A **Figura 29** ilustra a utilização do *plugin* no projeto desenvolvido.

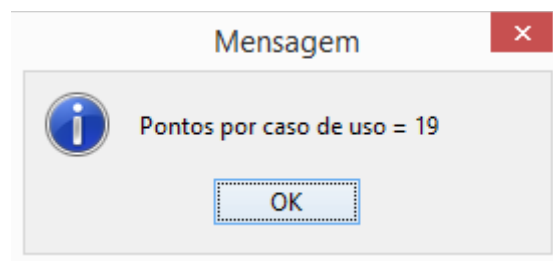


Figura 29: Resultado do *plugin*

O plugin desenvolvido possui algumas limitações, os atores deverão estar na pasta raiz do projeto, ou seja, não poderão estar dentro de nenhum pacote e os diagramas de sequência, obrigatoriamente devem estar dentro de um pacote nomeado como “diagramas de sequencia”, para que o plugin funcione corretamente.

### **Links**

Astah Professional – link para o download da ferramenta astah:

<http://astah.net/>

Link para download do Kit de Desenvolvimento de Software (SDK) do astah:

<http://astah.net/features/sdk>

Link para download do *plugin* desenvolvido:

<https://www.dropbox.com/sh/p7tv0mw2rak8dbh/AADUscRfesOaWFwC01mdaJmva?dl=0>

Link para tutoriais e documentação da API:

[http://astah.net/tutorials/plug-ins/plugin\\_tutorial\\_en/html/index.html](http://astah.net/tutorials/plug-ins/plugin_tutorial_en/html/index.html)